

ARCHITECTING APPLICATIONS FOR MULTIPLE UI TECHNOLOGIES

#SA3C, RIA: Flex/AJAX/AIR

Matt Woodward

Principal IT Specialist, United States Senate

www.mattwoodward.com/blog



CF.Objective()

What We'll Cover

- Architecting applications to support multiple user interface (UI) technologies
- Layered approach to architecture
- Separation of concerns in application layers
- Tight cohesion and loose coupling in objects
- Importance of the service layer

What We Won't Cover

- The basics of OO development
 - I'm happy to answer specific questions as we go, but I won't be covering any OO-101 type materials

Who Am I?

- I'll ignore the deeper philosophical implications and give you the basics ...
- Principal IT Specialist, Office of the Sergeant at Arms, United States Senate
- ColdFusion Developer since 1997
 - Also dabble in Flex and Java
- Mach-II Release Coordinator
- Manager, Capitol Hill User Group
- Member of the Open BlueDragon Steering Committee
- Adobe Community Expert for ColdFusion
- Co-Host, ColdFusion Weekly Podcast

Agenda

- Overview of layered OO architecture in CF applications
- Discussion of domain modeling and the importance of the business logic layer
- Importance of detaching your business logic from any particular UI or framework
- Examine a simple contact manager application

Layered Application Architecture

“There’s Pasta in My Software!”

CF

CF.Objective()

Spaghetti

- No matter how you try to dress it up, just plain nasty



Spaghetti with Meatballs

- Better, but still a little scary



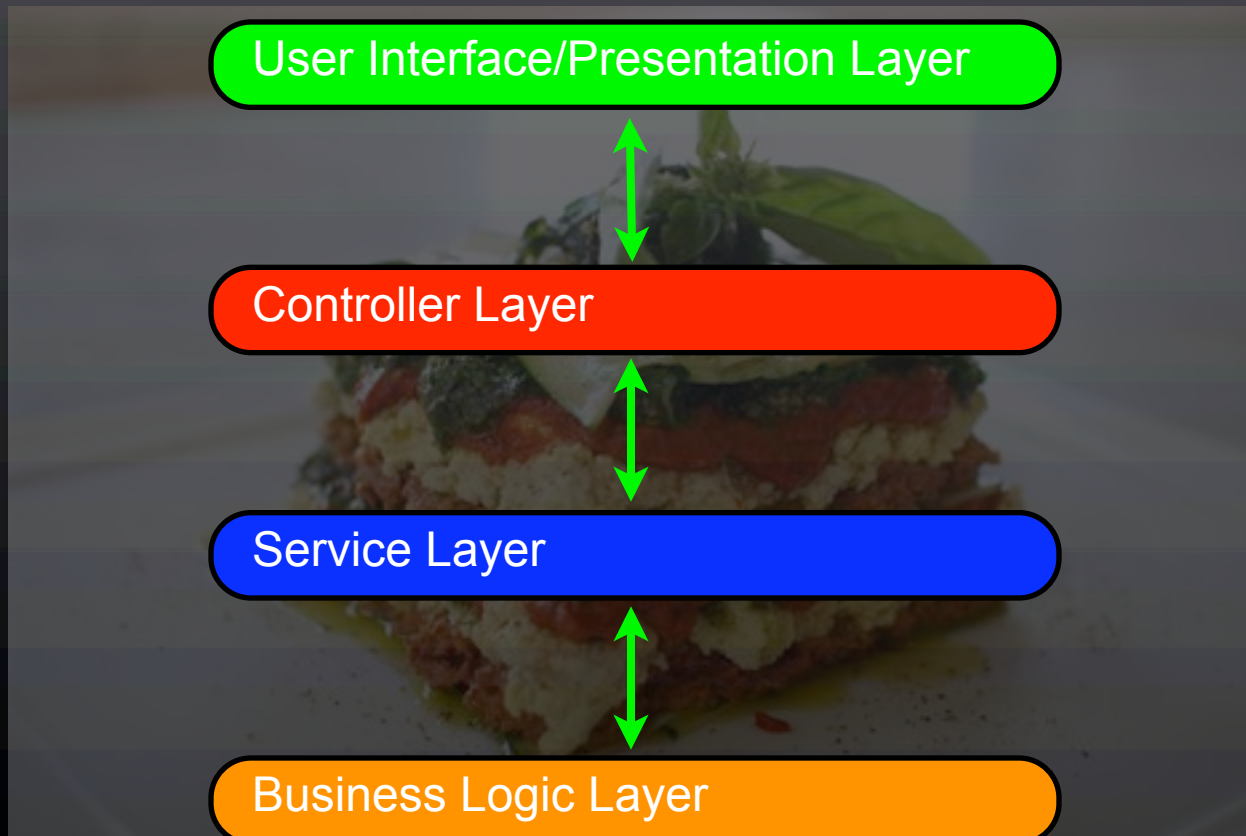
Spaghetti:
Now with CFCs!

Lasagne

- Elegant, refined ... the ultimate goal



The Layers (Mmmm ... Lasagne)

V**C****M****M**

Master Your Domain

No, that's not what I meant ...



CF.Objective()

The Domain Model

- Wikipedia: “... conceptual model of a system which describes the various entities involved in that system and their relationships”
- Objects, attributes, and methods
 - objects = nouns
 - methods = verbs
- Spend a lot of time here!
 - think before you code

Example Domain Model

- Contact management system
- Think about what objects are involved and how they relate to one another
- Use UML to think things through
 - easy to change things on a diagram
 - much harder to change (and you'll be more reluctant to change) once you write code

Where's the Database?



The Database Isn't Important

- OK, it *is* important, but not as important as some people think it is
- It's where you store your data, but it does NOT drive your application design
- On the application side, think in objects
- Worry about the database design once you have your domain model completed

Cohesion and Coupling

- VERY. IMPORTANT. CONCEPTS.
- Cohesion: the degree to which an object does one thing and does it well
- Coupling: the degree to which objects depend upon one another
- Strive for *high cohesion* and *loose coupling*

Peeling Back the Layers

Each one is more delicious than
the one before it

CF

CF.Objective()

Business Logic Layer

- This is the most important part of your application
- This is where you spend most of your development time
- Separate and protect it from the outside world
 - sheltered is good in this case!
- Make this work as reusable as possible

Business Logic Layer (cont.)

- Your business logic layer should not know or care who's talking to it (security concerns aside)
- Your business logic layer should NOT expose any details about HOW it does what it does
- Your business logic layer should only expose the bare minimum needed by the outside world to talk to it

Business Objects

- Tight cohesion and loose coupling mean you will likely have ...
 - beans (the “nouns” in your system)
 - data access objects (DAOs/Gateways)
 - potentially other business objects, UDFs, etc.
- Make sure each object does one thing and does it well
- Make sure each object doesn't have to know much about any other object

Quick Look at Business Objects

- Code samples of beans and data access objects

The Service Layer

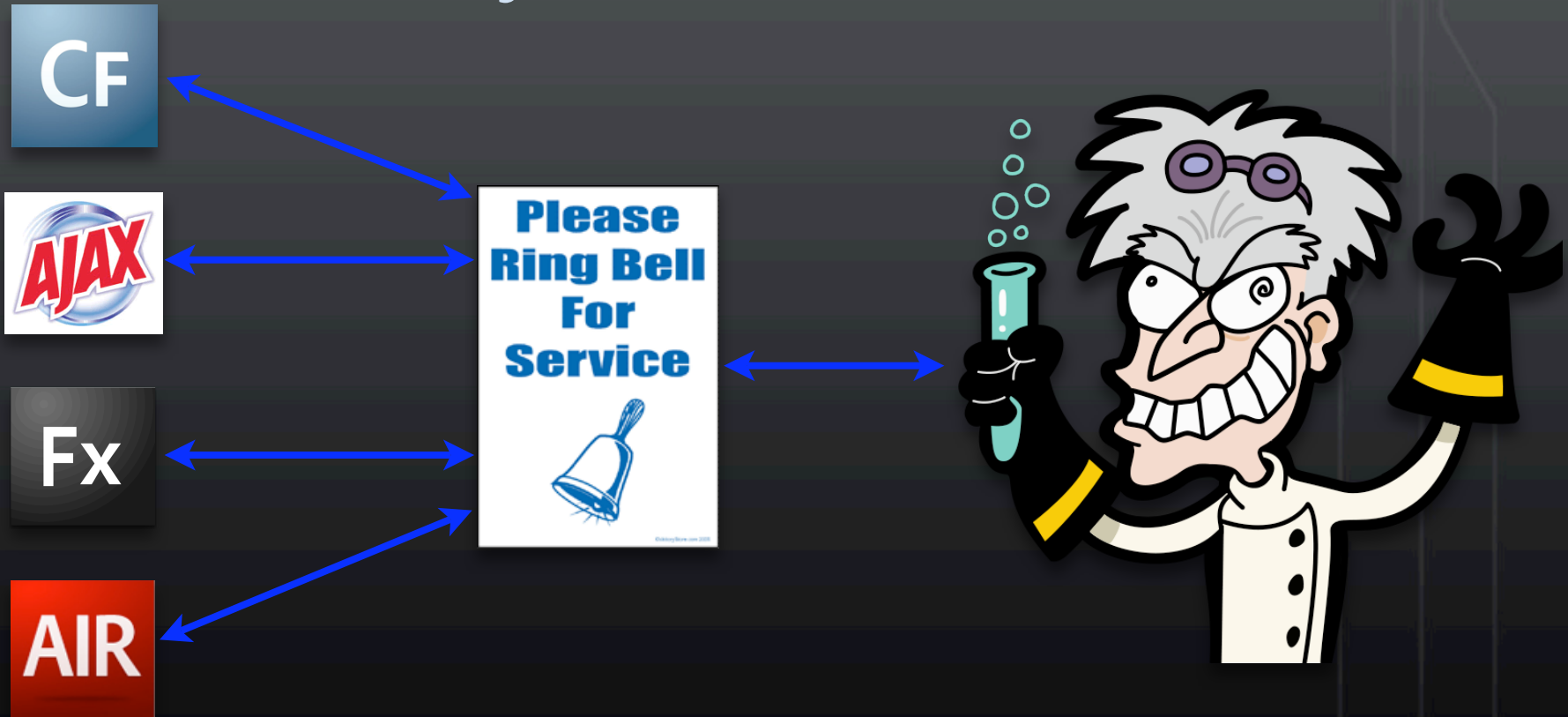
- Your business logic's receptionist



The Service Layer (cont.)

- If someone wants to talk to your business logic layer, they shouldn't have a direct line of communication
 - remember loose coupling--you don't want outside objects knowing too much
- Service layer is absolutely critical for the reusability of your business logic layer
- Think of the service layer as a public API to your business logic layer

Service Layer in Action



- You don't want anyone talking to your business logic directly

The Controller Layer

- We won't spend much time on this, but realize that there's actually one more layer involved
- Controller layer can be ...
 - roll your own
 - framework like Mach-II, Model-Glue, Fusebox
 - Flex is more or less its own controller, but there are also frameworks for Flex

The User Interface Layer

- If you've architected your application in a layered way, and used a service layer, you can pick your UI poison
- All UIs talk through their controllers to the service layer
- In the case of Flex, AJAX, web services, etc. it may make sense to have a remote proxy on top of the service layer

Remote Proxy

- Used to expose only certain methods from the service layer as remote
- Can better handle the specific needs of asynchronous calls from AJAX and Flex
- Keeps your service layer nice and clean
 - doesn't know or care who's talking to it

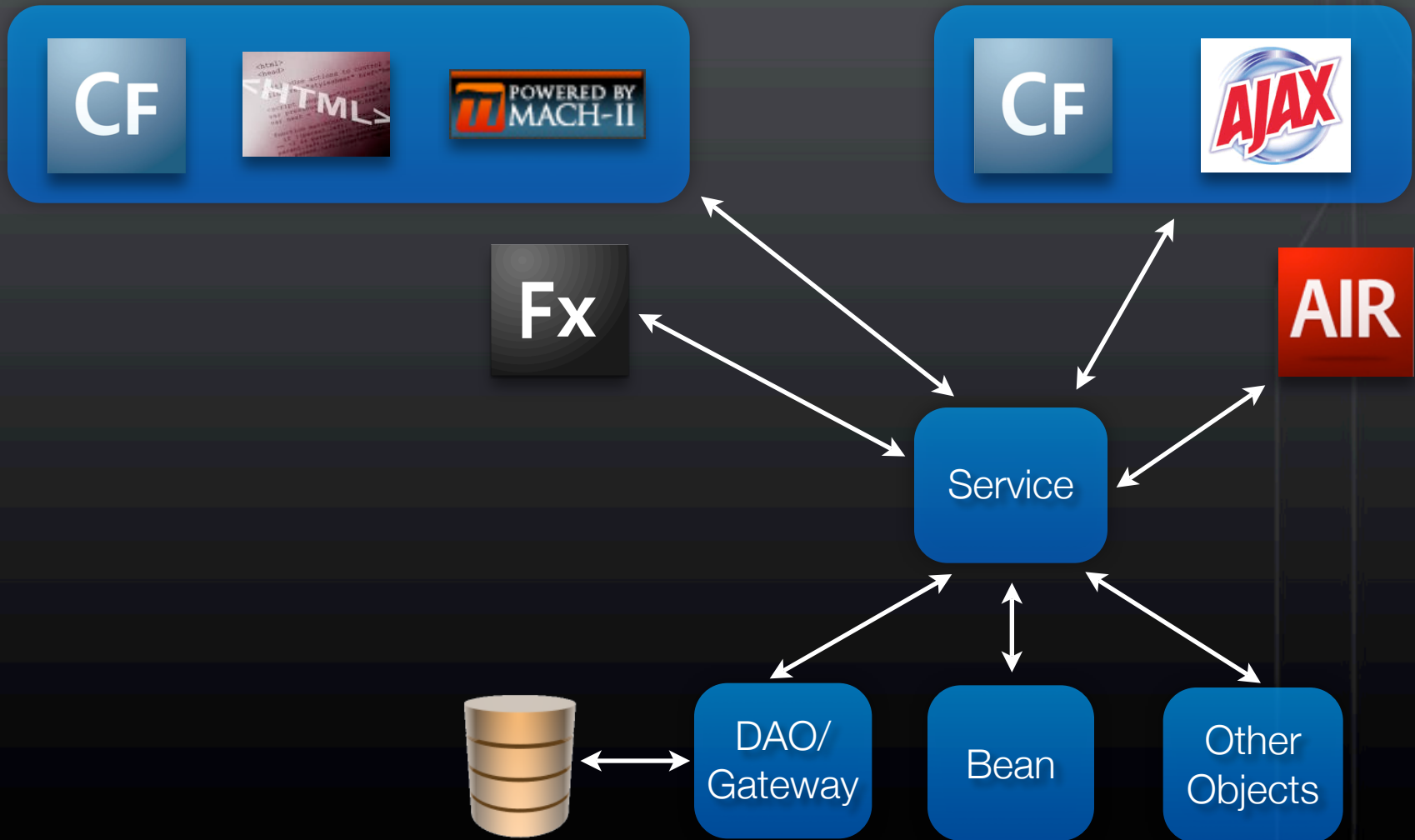
Putting It All Together

Where the Magic Happens

CF

CF.Objective()

Poetry in Motion



Contact Manager Application

- Demo and Code
- HTML, AJAX, and Flex all talking to the same business logic ***with no rework!***
- Example of a remote proxy used for Flex and AJAX

Recap and Q&A

CF

CF.Objective()

Recap

- With a little planning, architecting for multiple UIs is pretty simple
- Makes for good application architecture regardless of your UI technology choice
- Future-proof your application's business logic
- Remember:
 - spend time working on your domain model
 - separation of concerns
 - tight cohesion and loose coupling

Questions?

- Thanks!
- Matt Woodward
mpwoodward@gmail.com
<http://www.mattwoodward.com/blog>
<http://www.coldfusionweekly.com>